CLAIM AMENDMENTS

1.-15. (Canceled)

16. (Currently Amended) A method for routing or switching data packets, ~~including~~comprising the computer-implemented steps of:

receiving a data packet at an input interface on a router or switch;

looking up information in the header of said data packet in an expanded M-trie data structure, wherein said expanded M-trie data structure is organized as a multi-level tree including a root node, inferior nodes, and terminal nodes, wherein each node includes an address and an opcode; and

terminating said ~~lookup~~step of looking up information~~; and~~

~~routing said data packet at one or more output interfaces on said router or said switch~~.

17. (Canceled)

18. (Previously Presented) A method as in claim 16, wherein said opcode describes an operation to be performed that is based upon data included in a packet header, so as to facilitate a lookup of said packet header.

19. (Previously Presented) A method as in claim 16, wherein said address includes the address of a node in said expanded M-trie data structure that is to be traversed.

20. (Original) A method as in claim 16, wherein said expanded M-trie data structure includes a set of access control parameters.

21. (Previously Presented) A method as in claim 16, wherein said expanded M-trie data structure includes a set of Quality of Service (QoS) parameters.

22. (Previously Presented) A method as in claim 16, wherein said expanded M-trie data structure includes a set of Class of Service (CoS) parameters.

23. (Previously Presented) A method as in claim 16, wherein said nodes include opcodes for demultiplexing, opcodes for matching, and opcodes for hashing.

24. (Previously Presented) A method as in claim 23, wherein said opcodes for demultiplexing include instructions to demultiplex into branches of said expanded M-trie data structure based on contents of a byte of said packet header that is being read.

25. (Previously Presented) A method as in claim 23, wherein said opcodes for matching include instructions to compare the contents of a given byte of the flow label to given node data.

26. (Previously Presented) A method as in claim 23, wherein said opcodes for hashing include instructions to hash into different M-trie plus branches based on the contents of a given byte in said packet header.

27. (Canceled)

28. (Previously Presented) An apparatus for routing or switching data packets, comprising a device that performs a method comprising:

storing in memory an M-trie data structure, said data structure organized as a multi-level tree having a set of nodes, including a root node, inferior nodes and terminal nodes, wherein each node includes an address and an opcode;

receiving a data packet at an input interface on a router or switch, wherein the data packet includes information in at least a header with at least a field that is used by said M-trie data structure to indicate an action for said device to perform in order to select a leaf associated with said M-trie data structure;

looking up the information, wherein the looking up includes performing the action; and routing said data packet at one or more output interfaces on said router or said switch.

29. (Currently Amended) A method for routing or switching data packets, comprising the computer-implemented steps of:

storing in memory an M-trie data structure, said data structure organized as a multi-level tree having a set of nodes, including a root node, inferior nodes and terminal nodes, wherein each node includes an address and an opcode;

receiving a data packet at an input interface on a router or switch, wherein the data packet includes information in at least a header with at least a field that is used by said M-trie data structure to indicate an action for a router to perform in order to select a leaf associated with said M-trie data structure; and

looking up the information, wherein the looking up includes performing the action~~; and~~

~~routing said data packet at one or more output interfaces on said router or said switch~~.

30. (Previously Presented) A memory storing a program for performing a method for routing or switching data packets, comprising:

storing in memory an M-trie data structure, said data structure organized as a multi-level tree having a set of nodes, including a root node, inferior nodes and terminal nodes, wherein each node includes an address and an opcode;

receiving a data packet at an input interface on a router or switch, wherein the data packet includes information in at least a header with at least a field that is used by said M-trie data structure to indicate an action for a router to perform in order to select a leaf associated with said M-trie data structure;

looking up the information, wherein the looking up includes performing the action; and

routing said data packet at one or more output interfaces on said router or said switch.

31. (Canceled)

32. (Previously Presented) A memory as in claim 30, wherein said address includes an address of a node in said M-trie data structure that is to be traversed.

33. (Previously Presented) A memory as in claim 30, wherein said M-trie data structure includes a set of access control parameters.

34. (Previously Presented) A memory as in claim 30, wherein said M-trie data structure includes a set of Quality of Service (QoS) parameters.

35. (Previously Presented) A memory as in claim 30, wherein said expanded M-trie data structure includes a set of Class of Service (CoS) parameters.

36. (Previously Presented) A memory as in claim 30 wherein at least one of the root node, inferior nodes, or the terminal node includes an opcode for demultiplexing, an opcode for matching, and an opcode for hashing.

37. (Previously Presented) A memory as in claim 36 wherein said opcode for demultiplexing includes instructions to demultiplex into branches of the M-trie data structure based on contents of a byte of said packet header.

38. (Previously Presented) A method as in claim 36, wherein said opcode for matching includes instructions to compare the contents of a given byte of a flow label to given node data.

39. (Previously Presented) A method as in claim 36, wherein said opcode for hashing includes instructions to hash into different branches the M-trie data structure based on the contents of a given set of bytes in said packet header.

40. (New) A method as recited in Claim 16, further comprising routing said data packet at one or more output interfaces on said router or said switch.

41. (New) A method as recited in Claim 16, further comprising determining, based on one or more Access Control List (ACL) criteria stored in said expanded M-trie data structure, whether to drop or forward said data packet.

42. (New) A method as recited in Claim 41, wherein determining whether to drop or forward said data packet comprises matching said information in the header of said data packet to the one or more ACL criteria stored in said expanded M-trie data structure.

43. (New) A method as recited in Claim 41, wherein said one or more ACL criteria include at least one of a source address, destination address, and upper-layer protocol information.

44. (New) A method as recited in Claim 41, wherein said one or more ACL criteria are stored in a sub-tree of said expanded M-trie data structure.

45. (New) A method as recited in Claim 29, further comprising routing said data packet at one or more output interfaces on said router or said switch.

46. (New) A method as recited in Claim 29, further comprising determining, based on one or more Access Control List (ACL) criteria stored in said M-trie data structure, whether to drop or forward said data packet.

47. (New) A method as recited in Claim 46, wherein determining whether to drop or forward said data packet comprises matching said information to the one or more ACL criteria stored in said M-trie data structure.

48. (New) A method as recited in Claim 46, wherein said one or more ACL criteria include at least one of a source address, a destination address, and upper-layer protocol information.

49. (New) A method as recited in Claim 46, wherein said one or more ACL criteria are stored in a sub-tree of said M-trie data structure.